

CoinEasy

Kipp Martin
Booth School of Business
University of Chicago

November 14, 2011

IMPORTANT DISCLAIMER!

What follows are opinions of this author – **not official** opinions of the COIN-OR Strategic Leadership Board or the Technical Leadership Council.

I am speaking strictly as an individual COIN-OR user.

Background

- ▶ The idea for a “CoinEasy” came from a panel discussion at the 2009 San Diego INFORMS meeting.
- ▶ Purpose: get people up and running quickly.
- ▶ There is a prototype at:

<https://projects.coin-or.org/CoinEasy>

Background

Idea: characterize COIN-OR users by need and customize help for each group. A possible characterization:

1. A user who does **not** want to compile code – only run solvers. No compilers allowed.
2. A user who wishes to write applications that link to COIN-OR libraries, but do not want to compile project source code.
3. A user who wishes to compile COIN-OR project source code. My guess is that this is a small percentage of potential users.

Background

Opinion:

1. COIN-OR should make it easier for individuals in the first and second group.
2. COIN-OR should make it easier for users of Microsoft Windows.
3. Most project managers develop in some flavor of Unix so there is a natural bias.
4. There is a potentially large user base not familiar with concepts such as svn, configure, make, etc.

Group One

Option 1 Use a Commercial/Proprietary Modeling Language:

- ▶ **GAMS:** ships with COIN-OR solvers Bonmin, Clp, Cbc, Couenne, Ipopt, and OS (for remote calls).
- ▶ **MPL:** ships with COIN-OR solvers Clp and Cbc
- ▶ **AMPL:** Use **OSAmplClient** – must download this separately for the **CoinBinary** Web site.

If you are using GAMS, MPL, or AMPL, life doesn't change if you want to use COIN-OR. Just tell the modeling language to use a COIN-OR solver.

Group One

Option 1 (Modeling Language Continued): Use a modeling language.

AMPL: Use OSAmplClient. To AMPL this is like any other solver except that you can place calls to remote solver servers or just solve the problem locally.

```
model hs71.mod;
option solver OSAmplClient;
option OSAmplClient_options "solver SYMPHONY ";
solve;
```

OSAmplClient gives the user access to Bonmin, Cbc, Clp, Couenne, DyLP, Ipopt, SYMPHONY, and Vol.

Group One

OSAmplClient is part of the CoinAll or OS binary download.

With OSAmpClient there is no installation process. The user can put the executable in any desired location.

Should OSAmpClient be a separate download from CoinAll binary.

Group One

Option 1 (Modeling Language Continued): Use a COIN-OR modeling language. There are three choices.

- ▶ **Coopr (Pyomo)** – Python based
- ▶ **PuLP** – Python based
- ▶ **Cmpl** – an alternative to GAMS and AMPL and has it won algebraic scripting language.

These require a checkout and installation.

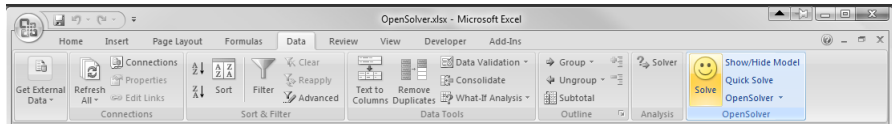
Actually, there is a fourth option, **FlopC++**, but these requires writing/compiling C++.

Group One

Option 2 (Excel Users): A user familiar with Frontline Solver can formulate a model as always using Solver.

Solve the model with COIN-OR Cbc using **OpenSolver**.

See www.opensolver.org.



There are no limits on model size using this option.

New Development: OpenSolver Studio build your models in Excel with PuLP and never leave Excel.

Group One

Option 3 (OSSolverService executable): A command line executable `OSSolverService` for reading problem instances (in OSiL format, AMPL `n1` format, or MPS format) and calling a solver either locally or on a remote server.

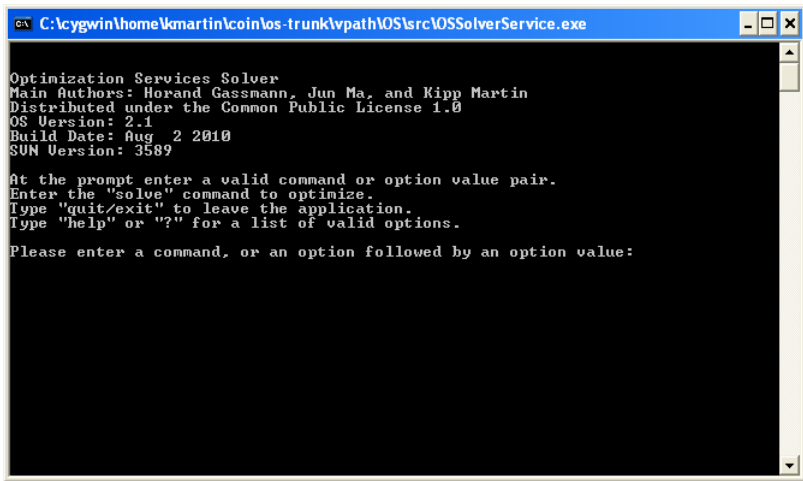
The `OSSolverService` has an interactive shell.

You can just “double-click” on the executable and it will guide you through the process.

Like `OSAmplClient` – nothing to install, download the binary and locate in any folder.

Group One

Option 3 (OSSolverService executable):



```
C:\cygwin\home\kmartin\coin\os-trunk\path\OS\src\OSSolverService.exe

Optimization Services Solver
Main Authors: Horand Gassmann, Jun Ma, and Kipp Martin
Distributed under the Common Public License 1.0
OS Version: 2.1
Build Date: Aug  2 2010
SUN Version: 3589

At the prompt enter a valid command or option value pair.
Enter the "solve" command to optimize.
Type "quit/exit" to leave the application.
Type "help" or "?" for a list of valid options.

Please enter a command, or an option followed by an option value:
```

Group One

If COIN-OR provided solvers running on a server the Group One users could use OSAmplClient or OSSolverService to access COIN-OR servers.

Nothing to build – just download the executable for the appropriate platform.

Should we try to do this?

Group Two – Use COIN-OR Libraries

Once again, a definition of group two:

- ▶ Want to build applications that link to COIN-OR project libraries.
- ▶ Do not need to build the projects from source.

Example: build a customized app to read data, build an instance, communicate instance to solver through and API, solve, present solution results to user.

Group Two – Use COIN-OR Libraries

Once again, a definition of group two:

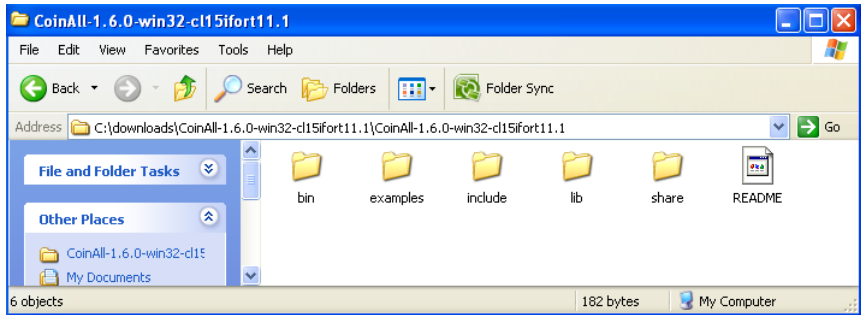
- ▶ Want to build applications that link to COIN-OR project libraries.

- ▶ Do not need to build the projects from source.

Example: build a customized app to read data, build an instance, communicate instance to solver through and API, solve, present solution results to user.

Group Two – Use COIN-OR Libraries

The CoinAll binary provides the following:



Group Two – Use COIN-OR Libraries

Time Out: Make sure we understand:

- ▶ **CoinEasy:** – a Wiki designed to provide information about using COIN-OR.
- ▶ **CoinAll:** – A macro project that contain many COIN-OR solvers including virtually all of the optimization solvers.
- ▶ **CoinBinary:** – A project (repository) devoted to distributing binaries of various COIN-OR projects

Group Two – Use COIN-OR Libraries

Examples folder – this is what is in **ApplicationTempates**

<https://projects.coin-or.org/CoinBazaar/wiki/Projects/ApplicationTemplates>

Objective: Provide code to illustrate the projects in CoinAll.

Objective Part II: Help Visual Studio users – configured project files are available.

Documentation also provided!

<https://projects.coin-or.org/svn/OS/trunk/OS/doc/UsingCoinAll.pdf>

Group Two – Use COIN-OR Libraries

The **examples** folder illustrates:

- ▶ How to take derivatives using algorithmic differentiation (CppAD). Useful when using nonlinear solvers.
- ▶ How to access numerous COIN-ALL solvers directly from code.
- ▶ How to build a model instance from code.
- ▶ How to access results (e.g. primal and dual solution information) from a solver.
- ▶ How to pass options to a solver using code.
- ▶ How to access and use the decomposition solver (Dip)
- ▶ How to access and use the branch-cut-price solver (Bcp)
- ▶ How to add cut generators from Cgl to Cbc.

Group Three – Use Project Source Code

- ▶ Try out new ideas – don't reinvent the wheel.
- ▶ Access to the source code allows a user to *make contributions to the project*.
- ▶ If the application does not behave as expected access to the source code allows a user to find bugs and suggest fixes.
- ▶ Access to the source code is educational. Implementation details of algorithms are typically not part of journal articles.
- ▶ The ability to duplicate results is crucial and helps promote high-quality research.

Group Three – Use Project Source Code

Option 1: Get the latest release at:

`http://www.coin-or.org/download/source/`.

Then do the Unix two-step:

- ▶ `configure`

- ▶ `make`

Option 2: Do an svn checkout. Live life in the fast lane – get a trunk version of a project.

Group Three – Use Project Source Code

Option 1: Get the latest release at <http://www.coin-or.org/download/source/>. Do the Unix two-step: **configure** and **make**.

Option 2: Do an svn checkout.

Need	Binary	Source Archive	Subversion
Solve a problem	X		
Link to a library	X		
Modify/use code		X	X
Project management			X

To Do:

Help graciously accepted:

- ▶ Adding more examples
- ▶ Document on how to use the examples
- ▶ The Wiki page

To Do:

Installers!!!

What else? Ideas?

Yet again, a COIN-OR server running solvers?